



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/518,974	12/21/2004	Richard Michael Taylor	5035-202US//P29,653	2918
20802	7590	09/18/2006	EXAMINER	
SYNNESTVEDT LECHNER & WOODBRIDGE LLP			HUISMAN, DAVID J	
P O BOX 592			ART UNIT	PAPER NUMBER
PRINCETON, NJ 08542-0592			2183	

DATE MAILED: 09/18/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 10/518,974	Applicant(s) TAYLOR, RICHARD MICHAEL	
	Examiner David J. Huisman	Art Unit 2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 21 December 2004.
- 2a) ☐ This action is FINAL. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-34 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-34 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☒ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 21 December 2004 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☒ All b) ☐ Some * c) ☐ None of:
1. ☒ Certified copies of the priority documents have been received.
 2. ☐ Certified copies of the priority documents have been received in Application No. _____.
 3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | 5) <input type="checkbox"/> Notice of Informal Patent Application |
| 3) <input checked="" type="checkbox"/> Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date <u>3/11/2005</u> | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

1. Claims 1-34 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: Preliminary Amendment as received on 12/21/2004 and IDS as received on 3/11/2005.

Specification

3. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.
4. The lengthy specification has not been checked to the extent necessary to determine the presence of all possible minor errors. Applicant's cooperation is requested in correcting any errors of which applicant may become aware in the specification.

Claim Objections

5. Please replace all occurrences of "minimise" and "utilise" (or any variation) with --minimize-- and --utilize-- (or the appropriate variation), respectively.
6. Claim 28 is objected to because of the following informalities: Please insert a period at the end of the claim. Appropriate correction is required.
7. Claim 30 is objected to because of the following informalities: Please replace "may set" with --may be set--. Appropriate correction is required.

Claim Rejections - 35 USC § 112

8. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

9. Claims 11, 15, 23-26, and 31 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

10. Claim 11 recites the limitation "the executable block" in lines 3-4. There is insufficient antecedent basis for this limitation in the claim.

11. Claim 15 recites the limitation "the executable block" in line 3. There is insufficient antecedent basis for this limitation in the claim.

12. Claim 23 recites the limitation "the committed phase" in lines 1-2. There is insufficient antecedent basis for this limitation in the claim.

13. Claim 26 recites the limitation "the executable block" in line 3. There is insufficient antecedent basis for this limitation in the claim.

14. Referring to claim 31, the examiner does not understand the metes and bounds of the claim. The use of the word "affect" in this context is unclear to the examiner and it is suggested that applicant clarify what exactly is happening to the branch instruction and how that relates to a logically later strand in the block. The examiner will interpret this claim to mean that an entry mechanism is used to affect a branch in a logically later strand in the block.

Claim Rejections - 35 USC § 102

15. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

16. Claims 1, 3-10, 12-14, 16-21, 23-28, 30-31, and 34 are rejected under 35 U.S.C. 102(e) as being anticipated by Kadambi, U.S. Patent No. 7,055,021.

17. Referring to claim 1, Kadambi has taught a method of instruction execution within a microprocessor whereby:

(a) a sequence of operations are divided into individual strands. See column 7, lines 57-60, and column 8, lines 1-6. It should be noted that a single microprocessor is divided into a number of virtual processors so that a number of strands may be executed. Strands may also be further divided into strands of instructions which have certain dependencies.

(b) the strands are numbered to provide an implicit logical time ordering. See column 10, lines 51-58, and column 11, lines 4-11. Note that a threads are marked with identification numbers. Clearly, the first thread created will be numbered with the first identification number. The second thread created will be given a second ID, and so on. Then, these threads are fetched one after another in round-robin fashion, which implies time ordering. That is, in round-robin, a first thread is fetched, then a second thread, and so on until the last thread, and then it starts over again.

Art Unit: 2183

(c) certain operations from different strands may be executed out-of-order with respect to their original sequential order. See the title and abstract.

(d) each strand has a predication state that determines whether certain operations from the strand should be completed. See column 12, line 64, to column 13, line 12, and column 21, lines 32-50. Note that there are two types of speculation: Speculation on a branch path and speculation when it comes to loading data. In both cases, if speculation was incorrect, instructions from the strand are not completed (they are flushed or executed with dummy values (operands set to zero), so they do not complete as intended).

18. Referring to claim 3, Kadambi has taught a method as described in claim 1. Kadambi has further taught that a plurality of strands are further composed into an executable code block.

Strands inherently include instructions and groups of instructions for an executable code block.

19. Referring to claim 4, Kadambi has taught a method as described in claim 3. Kadambi has further taught that means are provided to reset the predication status of each individual strand at the start of the execution of the code block. That is, the examiner deems this to be inherent, as when the system first begins execution of all of the strands, no speculation is occurring, and consequently, all operations are to complete normally. Only when speculation occurs will predicates be made invalid.

20. Referring to claim 5, Kadambi has taught a method as described in claim 4. Kadambi has further taught that certain operation strands can be given an abort status indicating that certain operations in the block could not be completed. Again, see column 12, line 64, to column 13, line 12, and column 21, lines 32-50. If misspeculation is realized, the strand is "given abort status" by indicating that certain operations cannot be completed, and so they are flushed.

21. Referring to claim 6, Kadambi has taught a method as described in claim 5. Kadambi has further taught that the abort status mechanism may be used to support the recovery from data speculative operations between strands. Clearly, by aborting instructions that are not to be completed, this is helping support the system to recover from misspeculation. After the incorrect instructions are flushed (or reexecuted), the system will again be on a correct execution path.

22. Referring to claim 7, Kadambi has taught a method as described in claim 6. Kadambi has further taught that execution of the code block should be repeated when an abort occurs. See column 21, lines 45-50.

23. Referring to claim 8, Kadambi has taught a method as described in claim 7. Kadambi has further taught that the predication status of individual strands is set upon a repeat execution such that strands that have already been completed are not re-executed. See column 21, lines 45-50. Instructions of strands that need to be reexecuted are reexecuted (i.e., the predication state of those instructions is set so that they are able to execute again). Instructions of strands that are not affected by the misspeculation do not need to be reexecuted, and therefore are not. That is why only dependent instructions of the speculation are repeated.

24. Referring to claim 9, Kadambi has taught a method as described in claim 3. Kadambi has further taught that the subdivision of code into strands is performed from an original sequential stream of instructions. See column 8, lines 1-6.

25. Referring to claim 10, Kadambi has taught a method as described in claim 3. Kadambi has further taught that the subdivision of code into executable blocks is performed from an original sequential stream of instructions. See column 8, lines 1-6.

Art Unit: 2183

26. Referring to claim 12, Kadambi has taught a method as described in claim 1. Kadambi has further taught that operations from the strand may be tagged within their execution word format to indicate the strand to which they belong. See column 10, lines 49-58.

27. Referring to claim 13, Kadambi has taught a method as described in claim 12. Kadambi has further taught that a tagged strand number is utilised in the control logic of the functional unit to affect the execution of the operation. See column 10, lines 49-63.

28. Referring to claim 14, Kadambi has taught a method as described in claim 13. Kadambi has further taught that execution from a disabled strand substantially disables the operation or prevents writeback of results that could affect processor state. See column 12, line 64, to column 13, line 12. Note that misspeculation on a branch causes the strand to be substantially disabled such that instructions are flushed (i.e., they are unable to complete and write results that affect processor state).

29. Referring to claim 16, Kadambi has taught a method as described in claim 14. Kadambi has further taught that the execution state of each operation strand is distributed to certain functional units by a global bus structure. As previously discussed, when misspeculation occurs, either a flush signal is sent to functional units or dummy operands and ultimately a replay signal are sent to functional units. These signals represent the execution state of the strands in that they indicate that they are in a incorrect execution state.

30. Referring to claim 17, Kadambi has taught a method as described in claim 16. Kadambi has further taught that the strand execution state is calculated and maintained in a strand control unit. This is deemed inherent as some subsystem must exist which will calculate when the

Art Unit: 2183

execution is correct and when it is incorrect, and based on the state, the appropriate signals will be sent.

31. Referring to claim 18, Kadambi has taught a method as described in claim 17. Kadambi has further taught that the strand control unit receives requests to modify strand status from one or more functional units. The strand control unit can only indicate that execution needs to be repeated or flushed in response to a signal that misspeculation has occurred. Consequently, the strand control unit, after receiving a signal that misspeculation has occurred (which inherently comes from some functional unit(s), will modify the strand status such that it must be repeated or flushed.

32. Referring to claim 19, Kadambi has taught a method as described in claim 7. Kadambi has further taught that an abort mechanism is utilised to provide a load speculation mechanism allowing memory loads to be executed earlier than a logically preceding store operation that may access the same address. See column 21, line 62, to column 22, line 4.

33. Referring to claim 20, Kadambi has taught a method as described in claim 19. Kadambi has further taught that the load speculation mechanism provides recovery from incorrect speculations by repeat execution of the executable block without the requirement for special compensation code. See column 21, line 56, to column 22, line 4. Incorrect speculations result in recovery (repeating execution). There is no mention of special compensation code to perform this. The instructions that need to be replayed are simply replayed.

34. Referring to claim 21, Kadambi has taught a method as described in claim 19. Kadambi has further taught that detection of incorrect load speculations is performed by an explicit functional unit that is used to compare the addresses being used by the load and store operations.

Art Unit: 2183

See column 21, line 64, to column 22, line 4. If overeager loads need to be replayed, and overeager loads have the same address as an older store, then it is inherent that some comparison of addresses is performed by a functional unit to determine that the addresses are the same.

35. Referring to claim 23, Kadambi has taught a method as described in claim 1. Kadambi has further taught that the entry to the committed phase of each strand is performed in the implicit logical time ordering of the strands. The examiner deems this inherent as, in general, strands executed after previous strands will be committed after the previous strands are committed. That is, the very first strand is not executed and then held off from being committed until after every other strand is committed.

36. Referring to claim 24, Kadambi has taught a method as described in claim 23. Kadambi has further taught that an abort of a certain operation strand also causes an abort of all strands which are logically later. This is clearly the case, as if a load is misspeculated, then all dependents off of that load will have to be replayed. Different strands would include these dependent operations. See column 21, lines 32-50.

37. Referring to claim 25, Kadambi has taught a method as described in claim 23. Kadambi has further taught that a branch executed from a particular operation strand causes all strands which are logically later to be disabled. See column 12, line 64, to column 13, line 12, and note that when a branch is mispredicted, all strands that have executed along that incorrect path will be disabled.

38. Referring to claim 26, Kadambi has taught a method as described in claim 25. Kadambi has further taught that branches may be issued out of their original sequential order but are suitably resolved and no actual branch is performed until the end of the executable block is

reached. This is the idea behind out-of-order processing. A branch may be the 100th instruction in the program but not be the 100th instruction executed. Furthermore, a branch always ends an executable block as it branches to a new block, so a branch is not performed until the end of the block.

39. Referring to claim 27, Kadambi has taught a method as described in claim 1. Kadambi has further taught that operations from different strands may be interspersed in the execution word for the purposes of improving code density. See column 13, lines 48-50. Essentially, with multiple threads, operations from each thread are mixed together to form an execution word, i.e., a group of instructions that are processed simultaneously.

40. Referring to claim 28, Kadambi has taught a method as described in claim 23. Kadambi has further taught that an explicit operation may be issued that disables a set of strands depending on whether they are logically the first being executed. In the case of a mispredicted branch or an overeager load, for instance, the strands that are first executed following those instructions will be canceled (flushed) or replayed. Therefore, a flush operation or replay operation would be issued.

41. Referring to claim 30, Kadambi has taught a method as described in claim 3. Kadambi has further taught that the execution status of strands upon entry to the executable block may set from a parameter provided by a preceding branch operation. Conditional branches contain parameters (condition field) which, depending on their value cause a target strand of instructions to be skipped or executed, and a fall-through strand to be skipped or executed. That is, if the branch is to be taken, depending on its condition code, then the execution status of the target strand is "execute" while the execution status of the fall-through strand is "don't execute".

Art Unit: 2183

42. Referring to claim 31, Kadambi has taught a method as described in claim 30. Kadambi has further taught that an entry mechanism may be used to affect a branch in a logically later strand in the block. See Fig.5, component 502, and column 8, lines 10-12, and note that Kadambi includes prediction logic, which has prediction entries. This will affect any branch that can be predicted, including those that are in logically later strands (those that follow initial strands).

43. Referring to claim 34, Kadambi has taught a microprocessor adapted to execute instructions using the method of claim 1. See the claim 1 rejection.

44. Claims 1-2, 11-15, and 34 are rejected under 35 U.S.C. 102(e) as being anticipated by Asato, U.S. Patent No. 6,289,442.

45. Referring to claim 1, Asato has taught a method of instruction execution within a microprocessor whereby:

(a) a sequence of operations are divided into individual strands. See Fig.6A and Fig.6B and note the different strands, each of which is associated with a different tag. For instance, a first strand includes all instructions having the tag 0X 0X 0X 0X.

(b) the strands are numbered to provide an implicit logical time ordering. Again, see Fig.6B. As new strands are introduced, the tags associated with the strands are modified. So, the first strand to execute is assigned a first identification number. The second strand will be given a second ID, and so on. The different IDs imply that different strands are being executed over time.

(c) certain operations from different strands may be executed out-of-order with respect to their original sequential order. See column 6, lines 10-15.

(d) each strand has a predication state that determines whether certain operations from the strand should be completed. See the abstract.

46. Referring to claim 2, Asato has taught a method as described in claim 1. Asato has further taught that operations in one operation strand are able to modify the predication status of another strand. Looking at Fig.6, the first strand would be all instructions having the tag 0X 0X 0X 0X, the second strand would be all instructions having the tag 0X 10 0X 0X, and the third strand would be all instructions having the tag 0X 11 0X 0X. The last instruction in the first strand is a branch instruction (A3). Consequently, instructions A4-A8 make up the fall-through strand while instructions B1-B3 make up the target strand. As is known, only one of the second and third strands will be able to fully execute. Which one does execute is dependent on the outcome of the branch operation A3 in the first strand. So, an operation in a first strand is able to modify the predication status of another strand.

47. Referring to claim 11, Asato has taught a method as described in claim 1. Asato has further taught that each operation strand is subdivided into a number of phases according to the type of operations that may be issued and operations that modify processor state that is visible outside of the executable block may only be executed in the final phase of each strand. Each strand can be seen as being divided into a number of phases. Since all strands end in a branch instruction (Fig.6B), then each strand has a pre-branch phase and a branch-phase. A branch instruction modifies processor state such that it controls which path the processor will execute, and it executes in a final phase of the strand (the branch phase).

Art Unit: 2183

48. Referring to claim 12, Asato has taught a method as described in claim 1. Asato has further taught that operations from the strand may be tagged within their execution word format to indicate the strand to which they belong. See Fig.6B.

49. Referring to claim 13, Asato has taught a method as described in claim 12. Asato has further taught that a tagged strand number is utilised in the control logic of the functional unit to affect the execution of the operation. See the abstract and Fig.7. Essentially, the tag of the correct path is broadcasted and compared with the tags of the two paths being executed. The strand having the tag that does not match the broadcasted tag is invalidated.

50. Referring to claim 14, Asato has taught a method as described in claim 13. Asato has further taught that execution from a disabled strand substantially disables the operation or prevents writeback of results that could affect processor state. See the abstract and Fig.7.

51. Referring to claim 15, Asato has taught a method as described in claim 14. Asato has further taught that the tagging of operations may be selective and need only necessarily apply to operations that affect processor state that is visible outside of the executable block. See Fig.6B. The tagging is selective, as the tag values can be assigned according to a user-specified algorithm. In addition, since all instruction affect a processor state, all operations are tagged.

52. Referring to claim 34, Asato has taught a microprocessor adapted to execute instructions using the method of claim 1. See the claim 1 rejection.

Claim Rejections - 35 USC § 103

53. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

Art Unit: 2183

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

54. Claim 22 is rejected under 35 U.S.C. 103(a) as being unpatentable over Kadambi in view of Hastings, U.S. Patent No. 5,193,180.

55. Referring to claim 22, Kadambi has taught a method as described in claim 21. Kadambi has not taught that address checks are inserted into the code strands as a result of insertion of such operations within a graph representation of the strand built at code generation time.

However, Hastings has taught such a concept. See column 3, lines 29-55, and note that prior to memory operations in strands, address check instructions are inserted which allow the system to detect array boundary violations and similar data errors. As a result, in order to ensure that memory access violations are avoided in Kadambi, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Kadambi to include the insertion of address checks in the code, as taught by Hastings.

56. Claim 29 is rejected under 35 U.S.C. 103(a) as being unpatentable over Kadambi.

57. Referring to claim 29, Kadambi has taught a method as described in claim 10. Kadambi has not taught that an operation strand mechanism is used to convert conditional blocks of code constructed using branches into separate operations strands that do not require a branch.

However, Official Notice is taken that the if-conversion and its benefits are well known and accepted in the art. That is, if-conversion is a process in which all branches are replaced with predication. Or, control dependence is changed into data dependence, which means that the code will no longer need to be executed using branch prediction which is time costly if there is ever a

Art Unit: 2183

misprediction. Instead, since operation execution merely depends on its predicate and its predicate is set by some preceding instruction, the instructions can simply be scheduled like every other out of order operation. There is no misprediction penalty to worry about, and furthermore, there would be no need for branch prediction hardware. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Kadambi such that branches could be replaced with predicated code.

58. Claims 32-33 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kadambi in view of Bala, U.S. Patent No. 6,233,678.

59. Referring to claim 32, Kadambi has taught a method as described in claim 1. Kadambi has not taught that the scheduling and construction of strands is influenced by profiling of the code. However, Bala has taught such a concept. See column 1, lines 12-47, and note that profiling is used to influence scheduling and recompile code to increase performance. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Kadambi to include a profiler as taught by Bala.

60. Referring to claim 33, Kadambi has taught a method as described in claim 1. Kadambi has further taught that a strand ordering is used to implement static speculations that provides performance benefit whilst seeking to minimise the chances of an incorrect speculation that requires recovery. This is inherent in any speculative, out of order machine as clearly the goal is to increase performance yet minimize costly incorrect speculation recovery.

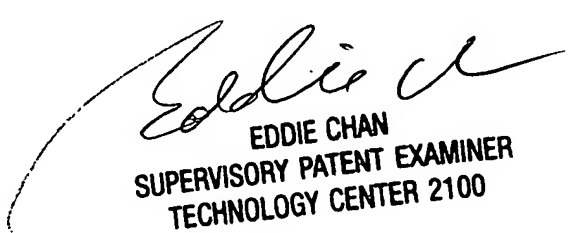
Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (571) 272-4168. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

DJH
David J. Huisman
September 13, 2006



EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100